State Machines

Exercise 1: Modify the diagram so the state machine counts to 11 and stops. Add a down input that cause the values to count down.

Exercise 2: Show the state transition diagram and table for a 2-bit counter with reset, enable, and down inputs. Reset should have priority. Write the Verilog.

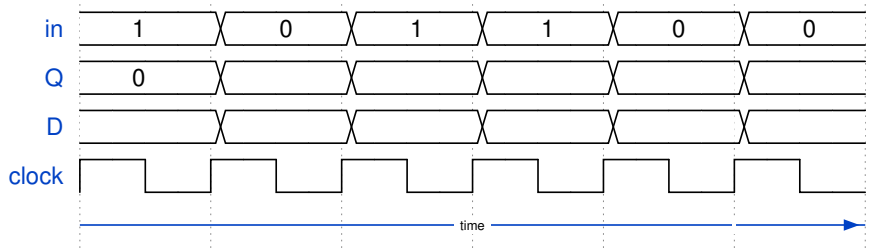
Exercise 3: What value of N would result in a 20 ms delay if the clock frequency is 50 MHz? How many bits are needed for this timer's register?

Exercise 4: Assume the timer above is reset to $N - 1$ each time it reaches 0. For how long is the register value 0? What are the period and frequency of a signal that is inverted each time the count reaches 0?

Exercise 5: Write the state transition table for the state machine for the **`lights`** output.

Exercise 6: The example above is an N-bit shift register that shifts the bits right. Draw a block diagram and write the Verilog for a 6-bit shift register that shifts left.

Exercise 7:



Fill in the diagram above for a 4-bit ($N = 4$) right-shift shift register. Assume the initial value is zero. Which bit is the oldest (first) value in the D waveform? Which bit of the shift register holds the oldest value?

Exercise 8: Draw a block diagram and write the Verilog for a circuit that sets an output named **`detect`** high when the sequence of values 1, 1, 0, 1 has appeared on an input named **`in`** on successive rising edges of the clock.

Exercise 9: For which states would **`falling`** be asserted? **`rising`**? Draw the schematic and write the Verilog for this state machine. Assume an input **`in`** and a 2-bit register **`bits`** that holds the two most recent input values.

Exercise 10: Write `always_ff` statements that implement these state machines.

Exercise 11: How could you modify the code so that **`digits`** is only updated when an **`enable`** input is asserted?

Exercise 12: How many states can this state machine have?

Exercise 13: Draw the state transition diagram for this simpler implementation. How many states are there? Write the Verilog using a 3-bit **`count`** state variable.

Exercise 14: How much logic is required to detect a state when a binary encoding is used? With a one-hot encoding?

Exercise 15: If we used 8-bits of state information, how many states could be represented? What if we used 8 bits of state but used a "one-hot" encoding?